The Heterogeneous Fleet Vehicle Routing Problem with Light Loads and Overtime: Formulation and Population Variable Neighbourhood Search with Adaptive Memory

Corresponding Author: Lina Simeonova

Affiliation: Centre of Logistics and Heuristic Optimisation (CLHO), Kent Business School
University of Kent, Canterbury, United Kingdom
Telephone: +44 7828198137
Fax: +44 1227 761187
Email: ls444@kentforlife.net

Co-Author: Niaz Wassan

Affiliation: Centre of Logistics and Heuristic Optimisation (CLHO), Kent Business School
University of Kent, Canterbury, United Kingdom
Telephone: +44 1227 823921
Fax: +44 1227 761187
Email: N.A.Wassan@kent.ac.uk

Co-Author: Said Salhi

Affiliation: Centre of Logistics and Heuristic Optimisation (CLHO), Kent Business School
University of Kent, Canterbury, United Kingdom
Telephone: +44 1227 824672
Fax: +44 1227 761187
Email: S.Salhi@kent.ac.uk

Co-Author: Gábor Nagy

Affiliation: Centre of Logistics and Heuristic Optimisation (CLHO), Kent Business School
University of Kent, Canterbury, United Kingdom
Telephone: +44 1227 827822
Fax: +44 1227 761187
Email: G.Nagy@kent.ac.uk

The Heterogeneous Fleet Vehicle Routing Problem with Light Loads and Overtime: Formulation and Population Variable Neighbourhood Search with Adaptive Memory

Lina Simeonova, Niaz Wassan, Said Salhi, Gábor Nagy
Centre for Logistics and Heuristic Optimisation (CLHO), Kent Business School
University of Kent, Canterbury, Kent

Abstract

In this paper we consider a real life Vehicle Routing Problem inspired by the gas delivery industry in the United Kingdom. The problem is characterized by heterogeneous vehicle fleet, demand-dependent service times, maximum allowable overtime and a special light load requirement. A mathematical formulation of the problem is developed and optimal solutions for small sized instances are found. A new learning-based Population Variable Neighbourhood Search algorithm is designed to address this real life logistic problem. To the best of our knowledge Adaptive Memory has not been hybridized with a classical iterative memoryless method. In this paper we devise and analyse empirically a new and effective hybridization search that considers both memory extraction and exploitation. In terms of practical implications, we show that on a daily basis up to 8% cost savings on average can be achieved when overtime and light load requirements are considered in the decision making process. Moreover, accommodating for allowable overtime has shown to yield 12% better average utilization of the driver's working hours and 12.5% better average utilization of the vehicle load, without a significant increase in running costs. We also further discuss some managerial insights and trade-offs.

Keywords

Real life Vehicle Routing; Population Variable Neighbourhood Search; Adaptive Memory; MIP Formulation; Managerial Insights

1. Introduction and Literature Review

The evolution of VRP variants is typically inspired by real life operations and there is a noticeable trend in the literature to bring VRP research closer to real life routing practice. There are many real life inspired applications of the VRP over the years, but it was not until 2006 when real life VRPs were presented as a class of the VRP family under the term *'rich'* VRPs (Gribkovskaia et al., 2006). They can also be referred to as *Multi-Attribute* VRPs (Vidal et al., 2014), *General* VRPs (Goel and Gruhn, 2008) or simply *real life* VRPs. Real life VRPs (RVRPs) proposed in the literature are very different from one another, and are usually not revisited by researchers with the same features. There is no universally accepted definition or consistent abbreviation for real life VRPs. For instance, Hasle, Løkketangen and Martello (2006) state that rich VRPs include aspects that are essential to the routing practice in real life, while Lahyani, Khemakhem and Semet (2015) suggest that the richness of the problems can stem from various attributes/constraints of the real life routing practice, either operational or strategic. Some authors introduce problem specific constraints such as outsourcing of vehicles (Stenger et al., 2013), customer prioritization constraints (Cornillier, 2009), specified times for cleaning vehicles (Oppen and Lokketangen, 2008) or environmental protection and Green VRPs (Erdogan and Miller-Hooks, 2012). Other authors such as, Archetti, Savelsbergh and Speranza (2016) introduce a RVRP with occasional drivers while Naji-Azimi et al. (2016) study a RVRP with desynchronized arrivals to the depot.

To the best of our knowledge there is no paper which considers a VRP relevant to the commercial gas delivery industry, incorporating the same real life features, namely light loads, demand-dependent service times and allowable overtime with unlimited fleet. However, there are some RVRPs which consider similar real life aspects, but from different perspectives. For instance, Zachariadis, Tarantillis and Kiranoudis (2015) tackle a load-dependent VRP with an iterative metaheuristic, while Nagy, Wassan and Salhi (2013) investigate a VRP with restricted mixing of the load using Reactive Tabu Search. Seixas and Mendes (2013) incorporate drivers working hours into a multiple trip VRP with heterogeneous fleet solved by Column Generation, whereas Battarra, Monaci and Vigo (2009) impose a shift length constraint for each vehicle in a minimum multiple trip VRP solved by a decomposition iterative heuristic with an adaptive guidance mechanism. Similar to the RVRP introduced here, Kok, Hans and Schutten (2012) consider time-dependent travel times where the aim is to reduce the impact of traffic congestions addressed by an adapted Dijkstra Algorithm and restricted dynamic programming.

Typically, papers considering allowable overtime are researched with fixed (limited) number of vehicles. For instance, Ren, Dessouky and Ordonez (2010) tackle a multi-shift problem with allowable overtime which is inspired by the healthcare industry and develop a problem specific algorithm namely the Shift Dependent heuristic. Moon, Lee and Seong (2012) propose a VRP with time windows, allowable overtime and outsourcing vehicles solved by a Genetic Algorithm and Simulated Annealing hybrid algorithm. This paper uses overtime with unlimited number of vehicles from each type, which raises an interesting trade-off between using the allowable overtime or using extra vehicles.

Most of the proposed methods for addressing RVRPs in the literature are heuristic-based. In this paper we use Adaptive Memory Procedure (AMP) as a method in its own right which is effectively hybridized in several novel ways with a population-based Variable Neighbourhood Search (VNS). The AM concept is first introduced by Rochat and Taillard (1995) as a complement to Tabu Search (TS) and refers to a special utilization of the memory during the search process. AM can be defined as a special data structure, which initializes a set of solutions and during the search process keeps track of the "best" components of the solutions, which are later combined to build better quality solutions (Tarantillis, 2005). In the VRP context, the use of AMP is still mostly as a complement to TS or other methods such as Particle Swarm Optimization (Yin, Glover and Laguna, 2010) and Path Relinking (Li, 2010), which also have embedded memory structures.

One of the most important methodological considerations regarding AMP is the way *"good"* solution components are extracted from the memory. Tarantillis and Kiranoudis (2002) proposed the BoneRoute method, where good solution sequences are referred to as bones. Each bone has length and frequency. The rationale is that good solution sequences appear in good, medium and low quality solutions, hence the higher the frequency of a bone, the better the chance it is a promising solution component. Other methods utilizing AMP include the Solutions' Elite Parts Search introduced by Tarantillis (2005) and the Multi-start AMP (Li, 2012). A recent paper by Matei et al. (2015) also uses features relevant to our methodology for population survival within a memetic algorithm with immigration techniques applied to the HVRP.

Many adaptations of the VNS introduced by Mladenovic and Hansen (1997) are used across the VRP domain. Some of its most recent applications include a Two-Level VNS for the Multiple Trip VRP with Backhauls (Wassan et al., 2017) and Ant Colony empowered VNS for the VRP with

simultaneous pickup and delivery (Kalayci and Kaya, 2016). Sze, Salhi and Wassan (2016) propose a hybridization of adaptive VNS and Large Neighbourhood Search for the classical VRP, which is later extended to the cumulative capacitated VRP with min-sum and min-max objectives (Sze, Salhi and Wassan, 2017). For more information on the new advances of VNS, we refer to Mladenovic et al. (2016) and for further detail on hybridization search, to the recent book on heuristics by Salhi (2017).

The contribution of this paper is as follows.

(i) We introduce a real life VRP, which is inspired by the gas delivery industry. It is characterized with heterogeneous vehicle fleet, maximum allowable overtime, a special light load requirement and demand-dependent service times.

(iii) We propose a new learning-based Population Variable Neighbourhood Search algorithm with Adaptive Memory (PVNS_AMP). We hybridize the Adaptive Memory principles with a local search method, where the memory aspect is incorporated in a long-term learning fashion within a memoryless, yet powerful metaheuristic such as VNS.

(ii) A mixed integer formulation is developed and tested on small sized instances, where optimal solutions or upper/ lower bounds can be found.

(iv) Interesting practical implications for more efficient and cost effective routing practice relevant to the RVRP are also put forward.

The rest of the paper is organised as follows. Section 2 presents a description of the problem and provides a mixed integer formulation of the RVRP, followed by our motivation to study the PVNS_AMP methodology in Section 3. Computational results and their analysis alongside some interesting practical implications are presented in Section 4. The final section summarizes our findings.

2. Problem Definition and Formulation

The proposed RVRP in this paper is inspired by a real life gas delivery company in the UK. The management is faced with challenges on a daily basis, regarding their routing practice. At present, there are three main aspects of the routing which cause inefficiencies for the company. We incorporate those into our problem and report possible savings and practical implications.

Light Load requirement

An important practical aspect of this particular problem is the light load requirement, because if the load on the vehicle is too heavy, some customers who live in areas such as steep hills or soft grounds may not be able to be accessed. Therefore, when a light load customer is serviced by a given vehicle, the remaining load on that vehicle needs to be lighter than a certain threshold level. The company does not have an efficient way of incorporating the light load aspect into their delivery schedule. At the moment, if a customer has a light load requirement, for simplicity it is manually added at the end of the vehicle route to ensure lighter load, which unfortunately can lead to significant inefficiencies in scheduling.

Allowable Overtime

Another key aspect that is strategically not taken into account is to incorporate overtime in advance. The current practice is to offer overtime to drivers towards the end of their regular time. This means that any remaining customers after the regular time will be served by the driver who agrees to perform overtime, without consideration of the routing efficiency. In addition, it is very common that drivers refuse overtime if they are not told in advance and this may lead to delays in delivery, unsatisfied customers and increased costs for the company.

Demand-dependent service times

Another aspect of our RVRP that needs to be mentioned is that only 150 litres of gas can be pumped into the customer tanks per minute, which renders the service times to vary depending on the demand size, hence the demand-dependent nature of the service time. In other words, the larger the customers' demand, the more time it will take the delivery driver to satisfy the demand, which can impact on the maximum number of customers a driver can visit in one planning period. In this study, we incorporate the above three attributes into the RVRP, in order to show improvements in the planning efficiency, as well as cost savings.

The RVRP is modelled on a complete directed graph $G = (N, A)$, where *N* is the set of customers $N = \{0,1,\ldots,n\}$ with 0 being the depot, and $A = \{(i, j) : i, j \in N, i \neq j\}$ is the set of arcs where each arc $(i, j) \in A$ has associated distance d_{ij} and time t_{ij} .There are *k* types of vehicles, each with a capacity Q_k , $k = \{1,\ldots,K\}$. Each vehicle is associated with a variable cost v_k based on how much

fuel a specific vehicle consumes, given the vehicle's average speed. The number of vehicles of each type is considered unlimited. The distance is Euclidean and the cost is proportionate to the distance travelled. The travel time t_{ij} takes into account the average speed of the vehicles, which according to the company records is approximately 30 mph. Each customer $i \in N$ has a known demand q_i, which is generated at random and a known service time s_i. The service time can be calculated by dividing the demand by the gas pumping rate of 150 litres per minute. Customers are divided into two types, regular *R(R ⊆N),* which can be serviced at any time during the delivery period, and light load L $(L \subseteq N,\ R \cap L = \varnothing,\ R \cup L = N)$. If a customer is considered to be light load $(i \in L)$, it means that it can only be serviced if the remaining load in the vehicle is less than a specified threshold level, c_k for $k = \{1,...,K\}$. In our case the maximum proportion of customers with light load requirement can be up to 20% of the total customers served. T is the maximum regular time for each vehicle route (7 hours and 20 min). Table 1 provides a summary of the problem specifications.

Table 1: RVRP Problem Specifications

Customer Coordinates	Golden et al. (1984)
Customer Demands q_i	Randomly Generated with Uniform Distribution [630,3950]
Vehicle Capacity	13050 litres (Type A), 20880 litres (Type B)
Average Speed	30 mph
Service time s_i	150 litres per minute
Variable cost per mile	0.36p (Type A), 0.48p (Type B)
Set of Light Load Customers L⊆ *N*	Randomly Chosen

Formulation

Decision Variables:

$x_{ijk} \in \{0,1\}$, with $x_{ijk} = 1$ if vehicle *k* travels along arc (i, j), 0 otherwise;

y_{ijk} is a non-negative continuous variable, which denotes the remaining load on a vehicle k, travelling along the arc (i, j);

z_{ij} is a non-negative continuous variable along the arc (i, j), which denotes the accumulated travel time upon arrival at node *j*;

$$\text{Minimize } Z = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{K} x_{ijk} d_{ij} v_k \qquad (1)$$

Subject to:

$$\sum_{i=0}^{n}\sum_{k=1}^{K} x_{ijk} = 1; \qquad (j = 1,\ldots,n); \qquad (2)$$

$$\sum_{j=0}^{n}\sum_{k=1}^{K} x_{ijk} = 1; \qquad (i = 1,\ldots,n); \qquad (3)$$

$$\sum_{i=0}^{n} x_{ipk} - \sum_{j=0}^{n} x_{pjk} = 0; \qquad (k = 1,\ldots,K), (p = 0,\ldots,n); \qquad (4)$$

$$\sum_{i=1}^{n} y_{ijk} - q_i = \sum_{i=1}^{n} y_{jik}; \qquad (j = 0,\ldots,n), (k = 1,\ldots,K); \qquad (5)$$

$$\sum_{k=1}^{K} y_{ijk} \le Q_k \sum_{k=1}^{K} x_{ijk}; \qquad (i \ne j = 0,\ldots,n); \qquad (6)$$

$$x_{ilk} \le 1 - \left((y_{ilk} - c_k)/Q_k\right); \qquad (i = 0,\ldots,R), (l = 1,\ldots,L), (k = 1,\ldots,K); \qquad (7)$$

$$\sum_{i=1}^{n} z_{ij} - \sum_{i=1}^{n} z_{ji} = \sum_{i=1}^{n}\sum_{k=1}^{K} x_{ijk}\left(t_{ij} + s_i\right); \qquad (j = 0,\ldots,n); \qquad (8)$$

$$z_{ij} \le T \sum_{k=1}^{K} x_{ijk}; \qquad (i \ne j = 0,\ldots,n); \qquad (9)$$

$$z_{0i} = t_{0i} \sum_{k=1}^{K} x_{0ik}; \qquad (i = 1,\ldots,n); \qquad (10)$$

$$z_{ij} \ge 0; \qquad (i \ne j = 0,\ldots,n); \qquad (11)$$

$$y_{ijk} \ge 0; \qquad (i \ne j = 0,\ldots,n), (k = 1,\ldots,K); \qquad (12)$$

$$x_{ijk} = \{0,1\}; \qquad (i \ne j = 0,\ldots,n), (k = 1,\ldots,K); \qquad (13)$$

The Objective Function (1) aims to minimize the total cost of travel. Constraints (2)-(3) state that each vehicle arrives at a customer location and leaves that customer location exactly once. Constraint (4) ensures the connectivity of the routes. Constraints (5)-(6) govern the commodity flow conservation and capacity restriction. Constraint (7) ensures that the light load customers

$i \in L$ will only be serviced if the remaining load on the vehicle is less than the specified threshold c_k. Constraints (8)-(10) govern the maximum time allowed for each vehicle trip. Constraints (11)-(12) guarantee that the decision variables y_{ijk} and z_{ij} are positive, where constraint (13) specifies the binary nature of the decision variable x_{ijk}. The MIP formulation has $(n+1)(2n+3)+3n+LK(R+1)$ constraints, $kn(n+1)$ binary variables and $n(n+1)+n(n-1)k$ continuous variables.

In the case where overtime is allowed, we make the following additions to the formulation. O is the maximum allowable overtime (4 hours and 30 min) and β is the variable cost of overtime which is 1.5 times higher than the cost of the regular time. A new variable a_k denotes the return time at the depot for each vehicle and o_k is a new decision variable denoting any overtime used. The variable z_{ij} is replaced by the variable z_{ik}, which represents the arrival time at customer i, for each vehicle $k = \{1,...,K\}$, where *M* is a significantly large constant. The following components of the original formulation need to be amended, in order to account for allowable overtime.

$$\text{Minimize } Z = \sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=1}^{K} x_{ijk} d_{ij} v_k + \beta \sum_{k=1}^{K} o_k \qquad (1a)$$

$$z_{jk} \geq z_{ik} + t_{ij} + s_j + M\left(\sum_{k=1}^{K} x_{ijk} - 1\right); \quad (i = 0,...,n), (j = 1,...,n), (k = 1,...,K); \qquad (8a)$$

$$a_k \geq z_{ik} + t_{i0} + s_i + M\left(x_{10k} - 1\right); \quad (i = 1,...,n), (k = 1,...,K); \qquad (9a)$$

$$a_k \leq T + O; \quad (k = 1,...,K); \qquad (10a)$$

$$o_k \geq a_k - T; \quad (k = 1,...,K); \qquad (11a)$$

$$a_k, z_{ik}, o_k \geq 0; \quad (k = 1,...,K); \qquad (12a)$$

The extended objective function equation (1a) refers to the total cost which includes the travel cost and the cost of any overtime used. It ensures that upon return to the depot any time over the maximum regular time T will be treated as overtime, multiplied by the overtime variable cost β and added to the total cost of travel. Note that β should not be too small, because if the penalty of overtime is very small it cannot influence the solution and it may be ignored. Constraints (8a)-(9a) denote the travel time upon arrival at customer i and the return time at the depot. Constraint

(10a) ensures the maximum travel time (including regular and overtime) is not exceeded. Constraint (11a) ensures that the allowable overtime occurs after the regular time, Constraint (12a) refers to the positive nature of the corresponding variables.

The number of vehicles can also become fixed to a certain number m by adding constrain (14), but the type of vehicle chosen remains variable. Moreover, if constraints (7)-(11) in the original model are relaxed, the RVRP reduces to the classical VRP Fleet Size and Mix.

$$\sum_{j=1}^{n} x_{0jk} = m; \qquad (k = 1, \ldots, K); \qquad (14)$$

3. The PVNS_AMP Algorithm

This paper adapts the classical form of VNS to Population VNS (PVNS) and enhances it with learning principles of AMP. We refer to the proposed method as PVNS_AMP. The main idea behind VNS is to explore successive neighbourhoods of the incumbent solution in depth, which provides intensification of the search process. In this paper a population based VNS is used, which means that more than one solution structure is kept into the memory and explored during the search. This is carried out for the purpose of diversification.

In contrast with the original AMP rationale, where memory initialization is done in advance, we perform it through learning. The learning takes place during the local search in Stage 1 of the algorithm, where promising solution sequences are memorized and evaluated based on their goodness of fit (solution quality). The recognition of good node sequences depends on their length and frequency, similar to the BoneRoute method, which is described in Section 2. However, the length of the node sequence is variable in our case. We refer to the extracted node sequences as *Elite Strings.* Moreover, a sequence of one node is also accepted, if it is a single customer route.

Another significant difference with previous AMP methods is that a node is allowed to be repeated in the extracted node sequences. The motivation behind it is that there may be more than one route composition and solution sequence, which could result in a best heuristic solution. Adjacency to the depot is recognised as well, which means that if a customer is best suited to be serviced first after the depot the *Elite String* can include the depot node. In addition, the PVNS_AMP has relatively few parameters, which are mostly related to memory extraction and

exploitation. The parameters used in this study are listed in Table 2. They have all been empirically tested and found most suitable for the RVRP problem.

Table 2: Parameters of the PVNS_AMP

Parameter	Description
P	Initial Solution Pool, which consists of 2 Initial Solutions $\mathrm{S} \in \mathrm{P}, \mathrm{S} = (1,...,2)$;
M	Memory Initialization Pool, which consists of a set of Neighbourhoods of S , with local optimum x'; *M* has variable size;
E	Memory Exploitation Pool with *Elite Strings*, which consists of Solutions survived to the PVNS_AMP Stage $\mathrm{E} \subseteq \mathrm{M}, \mathrm{E} = (\mathrm{S}'_1(\mathrm{x}'_1),...,\mathrm{S}'_{10}(\mathrm{x}'_{10}))$;
$\mathrm{iter}_{\max_1}$	Maximum iterations for Stage 1 is variable, until no further improvement for 2 consecutive iterations
$\mathrm{iter}_{\max_2}$	Maximum iterations for Stage 2 is 10
Elite Strings Recognition Criteria	A sequence of nodes is considered *Elite* if it has a Frequency >= 75% across all solutions saved in M
Proportion of *Elite Strings* in E	<=30% of the solution can be fixed by the *Elite Strings*
Elite Strings List	Variable Length

The PVNS_AMP consists of two stages. Stage 1 is called the *PVNS (Learning)* Stage, where information about the structure and the quality of the candidate solutions is gathered. At the end of the learning stage this information is used to recognize the *Elite Strings* which occur in more than 75% of the candidate solutions. Stage 2, which is the *PVNS_AMP* Stage, is the memory exploitation stage, where only the best 10 solutions, in terms of solution quality survive. The *Elite Strings* are encoded into the solutions which have survived from the previous stage and are further exploited using VNS until the best solution is found. Figure 1 provides a simple pseudo code for our PVNS_AMP algorithm. Each candidate solution S which enters the PVNS Stage after the initial solution generation has an objective function $F(x)$, which is the current local optimum, where $\mathrm{S}'(\mathrm{x}')$ denotes the best local optimum found during the local search of *S* in Stage 1. In Stage 2, the local search is further applied on each $\mathrm{S}'(\mathrm{x}')$ in the same fashion until no further improvement is found with $\mathrm{x}_{\mathrm{best}}$ representing the best solution found so far.

3.1. Stage 1 (PVNS Stage)

The purpose of Stage 1 is to construct the Memory Initialization Pool through past experience, as well as to compile knowledge about the solution space, which is then used to recognise the promising parts of the different candidate solutions.

Stage 1: PVNS Stage
Generate P **,** ***Set*** $M = \varnothing$
For Each $S \in P$
Do
Denote x as the local optimum of S , x' the current best local optimum
Set $x' = x, iter = 1;$
Do
Apply Neighbourhood Search Operators to S
if $F(x) < F(x')$ **, update** x'
Until no improvement of x
Add S' **to** M *, where* S' **=** S^{th} Neighbourhood of S associated with x'
Shake by probabilistic rules (explained in Section 3.1.)
While $iter < iter_{max_1}$
Next *S*
End of Stage 1

Elite String **Recognition** in M
Select $E \subseteq M$, $E = (S'_1(x'_1), ..., S'_{10}(x'_{10}))$

Stage 2: PVNS_AMP Stage
For Each $S' \in E$
Do
Denote x_{best} as the current local optimum
Set $x_{best} = x', iter = 1;$
Do
Apply Neighbourhood Search Operators to S'
if $F(x') < F(x'_{best})$ **, update** x'_{best}
Until no improvement of x'
Shake by probabilistic rules (explained in Section 3.2.)
While $iter < iter_{max_2}$
Next S'
End of Stage 2
End

Figure 1: Pseudo Code for the PVNS_AMP

Initial Solution Pool

There are two initial solutions, which make up the Initial Solution Pool. One is achieved through the Sweep Algorithm (Gillet and Miller, 1974) and the other is randomly generated. This is done for the purpose of diversification, so as to explore solutions with different topographical structures. Here the main rationale is that if an *Elite String* is recognized in a solution resulting from a construction heuristic and the same string re-appears during the search of a randomly generated solution, then there is a good chance the string is a promising component of the final heuristic solution.

Neighbourhood Search Operators

There are six Neighbourhood Search Operators used to explore a solution. The *1-1 intra-route swap*, exchanges the positions of each node with all other nodes on the same route. *1-0* and *2-0 inter-route shift* insert each one / two consecutive nodes respectively, in all feasible locations on all other routes. The *1-1 inter-route swap*, exchanges the positions of one node with all other nodes from all other routes; *2-1 inter-route swap*, exchanges the positions of 2 consecutive customers from one route with one customer from all other routes and *2-2 inter-route swap*, exchanges the positions of two consecutive nodes from one route with two consecutive nodes from all other routes.

It is believed that randomly generated solutions can be computationally expensive to turn into better quality solutions, especially if the best-improvement strategy is used. Therefore, the first improvement strategy is used in order to find immediate good links between nodes, hence speed up the learning process for the composition of the *Elite Strings*. All operators are used in a systematic order, where all feasible shifts and swaps are considered until no further improvement is found. The current best solution is saved into the Memory Initialization Pool after each iteration. Then the shake stage of VNS takes place, which is done by probabilistic rules. In Stage 1 two random customers from random routes are inserted into different routes at a random position. The reason for choosing a more vigorous shake is because when the solution enters the neighbourhood operators at the next iteration any good immediate links between nodes could re-appear if they were broken during the shake and the frequency of the link is likely to increase.

3.2. Stage 2 (PVNS_AMP Stage)

Stage 2 is the memory exploitation stage, where the knowledge gathered in Stage 1 (PVNS stage) is used to improve the solution quality. After Stage 1, the *Elite Strings* are recognised, according to the pre-defined criteria. The Memory Initialization Pool is then reduced to the best 10 candidate solutions in terms of solution quality and the *Elite Strings* are encoded into them. The *Elite Strings* become the fixed part of the solution structure, which does not change during further neighbourhood search. The remaining nodes remain a variable part of the solution. The Elite Strings List is of dynamic length. This is because in the different data instances, different number of *Elite Strings* can be recognised from the Memory Initialization Pool, which have a frequency of

75% and higher. When encoding the *Elite Strings* into the solutions, those with the highest frequency have priority. However, the proportion of *Elite Strings* which are encoded into a solution is limited to up to 30%. These solutions enter the second stage in a systematic fashion in ascending order in terms of solution quality. The operators and execution of the VNS search is the same as in Stage 1, but only the variable part of the solution is modified via the shift and swap operators. Having a proportion of the solution which remains fixed, acts as a neighbourhood reduction technique and speeds up the CPU time of the operators. The *Elite Strings* remain fixed during the shake stage as well. However, in Stage 2 of the algorithm, the shake does not distort the solution too much, where only one customer is randomly reassigned to a different route. This provides intensification of the search, but keeps the focus of the search in better regions. The population-based nature of the VNS (the survival of a number of candidate solutions) in the second stage is very important for diversification. The candidate solutions which enter the second stage of the algorithm are quite diverse in terms of solution structure; hence they contain different *Elite Strings* and provide for a better coverage of good local optima.

Elite Strings Encoding

The proportion of *Elite Strings* incorporated into the solutions in the second stage of the algorithm is an important methodological consideration. There is a clear trade-off between the proportion of the solution that is fixed via *Elite Strings*, solution quality, and computational time. If a smaller proportion of the solution is fixed, then the solution quality may not improve in the second stage as it is not focused enough into better search areas. Similarly, if too much of the solution is fixed, the *Elite Strings* may not in fact be elite, which can lead the search to explore a region that is falsely recognised as good. Also, the computational time decreases as the proportion of *Elite Strings* increases in the solution. Figure 3 illustrates this trade-off and shows that when the solution contains 30% or less *Elite Strings*, is sufficient for good memory exploitation. Another interesting observation is the fluctuation of the solution quality at different levels of *Elite Strings* encoding. It only fluctuates less than 4%, which suggests good quality extraction of *Elite Strings* even with up to 60% coverage of the solution. The example portrayed in Figure 2 is an instance with 100 customers with and without overtime. Both versions are graphically represented in order to show consistency in the algorithmic behaviour.

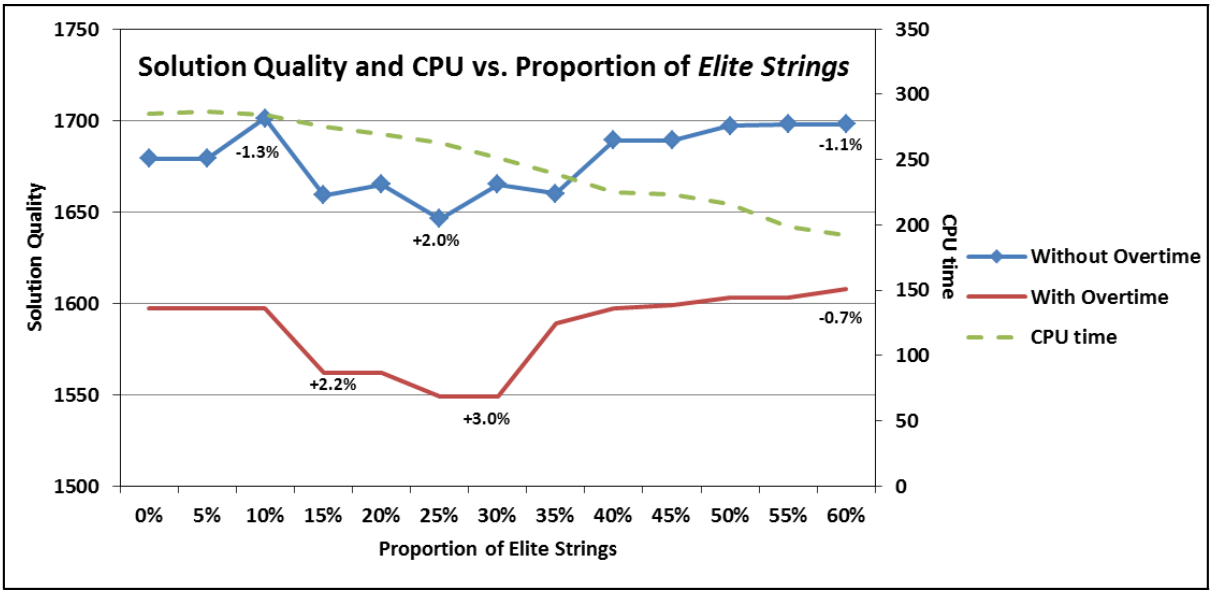

Figure 2: Instance *N = 100, L = 10%*

The population nature of the algorithm has another benefit when it comes to *Elite Strings* encoding. During the computational experience we found that it is possible to recognize a solution sequence as an *Elite String*, but in fact it is not an *Elite String*. Therefore, working with a population of candidate solutions allows for overcoming this possible drawback. This issue is illustrated in Figure 3, which shows two candidate solutions which survive from Stage 1 for an instance with 20 customers. The string **8-7-6** which is recognised as elite is present in the candidate solution with objective function of 501.6. However, looking at the optimal solution obtained from Cplex, the string 8-7-6 is not part of the optimal solution, hence it is not elite. Therefore, having a population based VNS, where there is a pool of candidate solutions with different solution structure and different *Elite Strings* is necessary.

Learning Mechanism

The PVNS_AMP hybridizes AMP with a memoryless method, which is not a common practice in the VRP domain. Therefore, it is important to show the benefit of learning and memory exploitation. Figure 4 shows that there is a benefit from using the AMP as a learning strategy for VNS. It is clear from the figure that the solution quality from the PVNS Stage fluctuates more during the runtime of the algorithm, whereas in the PVNS_AMP stage it is more stable and more focused in lower topography. This is because the fixed part of the solution (i.e. the *Elite String)*, is guiding the search to remain in better regions.

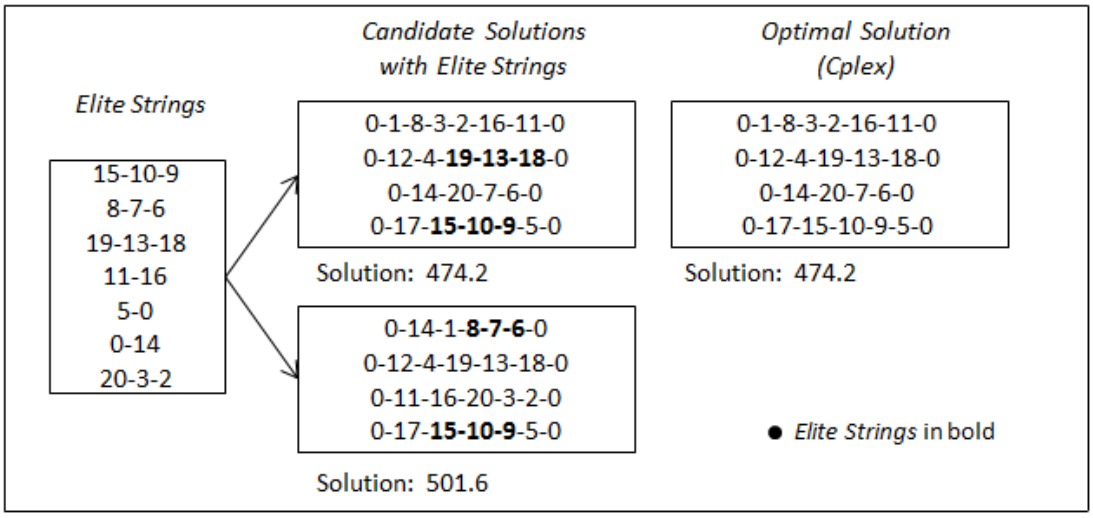

Figure 3: Encoding of Elite Strings into candidate solutions

The instance portrayed in Figure 4 shows another interesting observation. The candidate solutions S' enter the PVNS_AMP stage in ascending order based on their objective function value. It can be seen that the best solution in the PVNS_AMP stage was reached towards the end of the running time of the algorithm. This means that it was reached by a candidate solution $S' \in E$ with larger objective function value. This is an important observation when it comes to problems with overtime. The neighbourhood operators we use to explore the candidate solutions involve shifting a maximum of two customers at a time. This raises an interesting trade-off between using more allowable overtime vs. using an extra vehicle. Having a population VNS allows for the exploration of solutions which favour overtime, as well as solutions which favour an extra vehicle. The diversity of the solution pool means greater coverage of the search space, where not only the best cost solution is further explored, but also those with larger objective function.

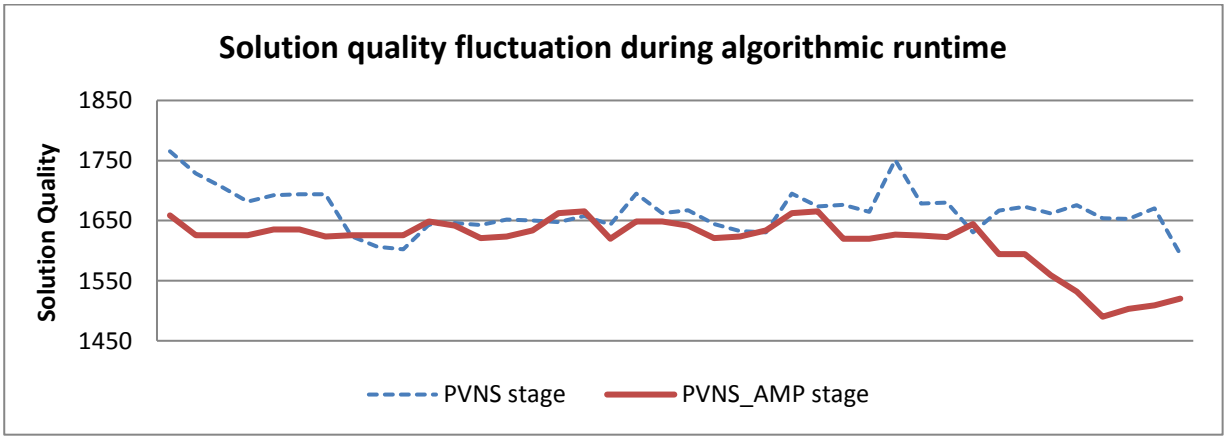

Figure 4: The RVRP with Overtime *N = 100, L = 10%*

4. Computational Results and Analysis

The PVNS_AMP is coded in C++ and all experiments are conducted on a PC with Intel Core i7 with 3.4 GHz. There are no standard benchmark instances for RVRPs, as they are tailored to real life practices, making it more difficult to comment on the algorithmic comparability aspect. It is common for researches addressing RVRPs to use randomly generated data, data provided by a company or adapted literature benchmark instances. We chose to adapt the problem instances of Golden et al. (1984). The original coordinates of the instances are used, where the other specifications for demand, vehicle capacity, average vehicle speed, service time and variable cost are informed by a real life gas delivery company and are detailed in Section 2.

4.1. The RVRP without Overtime

The RVRP is first solved using the MIP formulation provided in Section 3 in Cplex Version 12.6. The results are then compared to those from the proposed PVNS_AMP. Table 3 shows the results produced by Cplex and those by the PVNS_AMP, with the corresponding CPU times. The total CPU time (TCPU) is reported, as well as the time to the best found solution (BCPU). The last column shows the percent improvement in solution quality when AMP is incorporated into the PVNS.

There are a few observations that can be made from Table 3. First, incorporating AMP within VNS and exploiting the memory in Stage 2, leads to up to 5.2% improvement in the objective function. Second, looking at the TCPU for both stages, it can be seen that generally the higher the proportion of light load customers, the smaller the TCPU. This is a valid observation, because the higher the number of light load customers, the smaller the search space becomes, which restricts the local search allowable moves. The BCPU confirms the observation made in Figure 4, that some of the best solutions are found towards the end of the total runtime for both stages of the algorithm. This emphasizes on the benefit of using a population VNS.

To show the benefit of using the PVNS_AMP hybrid we have also performed some further testing using different versions of our algorithm. Table 4 shows the results achieved by using the classical VNS with the same algorithmic steps as detailed for Stage 1 of the PVNS_AMP, but applied to one candidate solution only, since the classical VNS applies local search to a single solution. Some interesting observations can be noted from Table 4. First, using a Population VNS results in up to 4% improvement of the solution quality on the RVRP without overtime, which shows the benefit

of using a pool of solutions. Second, there is up to 3% improvement on the classical VNS when AMP is incorporated within it. However, comparing the results from the VNS_AMP and the PVNS only, it can be seen that the PVNS performs slightly better than the VNS_AMP. One reason for this would be that the AMP needs a more diversified solution pool in order to be able to extract promising solution sequences, rather than only learning from one solution structure. Hence the PVNS_AMP has superior performance, with up to 8% improvement of the solution quality from the classical VNS. The average performance of the methods is also reported in Table 4.

It can be seen from Table 4 that the VNS_AMP results in improvement from the VNS, which means that a greater degree of intensification of the search space can improve the performance of the method. Therefore, we have tested the PVNS_AMP with more intensified local search, by increasing the number of iterations for each candidate solution for Stage 1 and Stage 2 of the algorithm. We set the number of iterations to 20, which was empirically found to be suitable. Additionally, the PVNS_AMP has a degree of randomisation and in order to have a more thorough testing we also have recorded the average results of the PVNS_AMP over 10 runs. These results are shown in Table 5. Testing the PVNS_AMP using 10 runs, with different starting seed shows some improvements on the larger sized instances with up to 1.03%, whereas giving more depth to the search with more iterations leads to an improvement of up to 1.45%. This shows that having a high degree of diversification is needed for good extraction of *Elite Strings*, but also shows an improvement when the local search is further intensified, emphasizing on the importance of the fine balance between diversification and intensification aspects of heuristic methods.

Table 3: Computational Results for the RVRP without overtime

N	L (%)[a]	CPLEX				Stage 1 (PVNS Stage)				Stage 2 (AMP Stage)				
		LB/ Optimal[b]	UB	Time[c]	Fleet Composition	Solution	TCPU[d]	BCPU[e]	Fleet Composition	Solution	TCPU	BCPU	Fleet Composition	IMP
20	10	**446.2**	-	4	3A 1B	**446.2**	3	<1	3A 1B	**446.2**	2	<1	3A 1B	0.00%
20	15	**446.9**	-	3	3A 1B	**446.9**	3	<1	3A 1B	**446.9**	2	<1	3A 1B	0.00%
20	20	**462.3**	-	3	3A 1B	**462.3**	2	<1	3A 1B	**462.3**	1	<1	3A 1B	0.00%
30	10	**560.1**	-	640	2A 3B	569.3	5	2	2A 3B	**560.1**	2	<1	2A 3B	1.62%
30	15	**560.1**	-	640	2A 3B	569.3	5	2	2A 3B	**560.1**	2	<1	2A 3B	1.62%
30	20	535.9	575.4	375[m]	-	565.3	4	2	2A 3B	565.3	3	<1	2A 3B	0.00%
50	10	701.1	901	1830[m]	-	879.1	16	5	8A 2B	852.2	10	5	6A 2B	3.06%
50	15	706.8	958.2	248[m]	-	882.3	15	10	5A 5B	867.2	10	5	4A 5B	1.71%
50	20	699.4	N/A	1109[m]	-	903.9	13	7	6A 4B	877.4	8	3	6A 4B	2.93%
75	10	993.1	1541	971[m]	-	1269.5	36	17	2A 8B	1244.1	25	20	2A 8B	2.00%
75	15	985.9	1391	1658[m]	-	1272.1	33	8	7A 5B	1254.3	22	15	6A 5B	1.40%
75	20	985.9	N/A	2662[m]	-	1292.4	31	15	8A 6B	1267.5	19	7	8A 6B	1.93%
100	10	1274.6	2908	1396[m]	-	1667.6	75	20	13A 5B	1646.4	55	48	13A 5B	1.27%
100	15	1248.4	2844	1930[m]	-	1744.1	79	36	13A 6B	1689.9	52	50	12A 6B	3.11%
100	20	1247.4	N/A	322[m]	-	1798.3	62	62	9A 9B	1705.3	40	14	10A 8B	5.17%

[a] L: Percent of $L \subseteq N$

[b] LB / Optimal: Optimal Solution in bold

[c] Time: Cplex computational time in minutes

[d] TCPU: Total runtime in seconds for the corresponding version

[e] BCPU: Time to best found solution in seconds for the corresponding stage

IMP: Percent Improvement with AMP

[m] Time recorded until system is Out of Memory

Table 4: Performance of different versions of the proposed algorithm on the RVRP without overtime

N	L	VNS[a]		PVNS[b]			VNS_AMP[c]			PVNS_AMP[d]		
		Solution	CPU	Solution	CPU	IMP	Solution	CPU	IMP	Solution	CPU	IMP
20	10%	446.2	<1	446.2	3	0.00%	446.2	<1	0.00%	446.2	5	0.00%
20	15%	446.9	<1	446.9	3	0.00%	446.9	<1	0.00%	446.9	5	0.00%
20	20%	462.3	<1	462.3	2	0.00%	462.3	<1	0.00%	462.3	3	0.00%
30	10%	569.3	<1	569.3	5	0.00%	569.3	3	0.00%	560.1	7	1.62%
30	15%	569.3	<1	569.3	5	0.00%	569.3	3	0.00%	560.1	7	1.62%
30	20%	565.3	<1	565.3	4	0.00%	565.3	3	0.00%	565.3	7	0.00%
50	10%	905.03	5	879.1	16	2.95%	893.1	11	1.34%	852.2	26	5.84%
50	15%	912.21	5	882.3	15	3.39%	903.9	7	0.92%	867.2	25	4.93%
50	20%	926.45	5	903.9	13	2.49%	926.4	7	0.01%	877.4	21	5.29%
75	10%	1319.62	11	1269.5	36	3.95%	1282.6	16	2.98%	1244.1	61	5.72%
75	15%	1324.1	10	1272.1	33	4.09%	1292.5	15	2.44%	1254.3	55	5.27%
75	20%	1340.7	10	1292.4	31	3.74%	1305.2	15	2.72%	1267.5	50	5.46%
100	10%	1734.2	18	1667.6	75	3.99%	1689.2	28	2.66%	1646.4	130	5.06%
100	15%	1816.9	17	1744.1	79	4.17%	1766.3	28	2.86%	1689.9	131	7.00%
100	20%	1859.4	15	1798.3	62	3.40%	1810.1	26	2.72%	1705.3	102	8.29%
Average		**1013.19**	**10.7**	**984.57**	**25.5**	**2.14%**	**995.24**	**13.5**	**1.24%**	**963.01**	**42.3**	**3.74%**

[a] Classical VNS without AMP

[b] Population VNS without AMP

[c] Classical VNS with AMP

[d] PVNS with AMP

IMP: Improvement from classical VNS

CPU time in seconds

Average: Solution, Time and IMP

Table 5: Further testing of the PVNS_AMP algorithm on the RVRP without overtime

N	L	PVNS_AMP[a]		PVNS_AMP[b]				PVNS_AMP[c]		
		Solution	CPU	Best Solution	Average Solution	Average CPU	% IMP	Solution	CPU	% IMP
20	10%	446.2	5	446.2	446.2	5	0.00%	446.2	9	0.00%
20	15%	446.9	5	446.9	446.9	5	0.00%	446.9	9	0.00%
20	20%	462.3	3	462.3	462.3	4	0.00%	462.3	9	0.00%
30	10%	560.1	7	560.1	560.1	10	0.00%	560.1	25	0.00%
30	15%	560.1	7	560.1	560.1	9	0.00%	560.1	23	0.00%
30	20%	565.3	7	565.3	565.3	9	0.00%	565.3	25	0.00%
50	10%	852.2	26	852.2	852.2	31	0.00%	852.2	79	0.00%
50	15%	867.2	25	867.2	867.2	32	0.00%	867.2	76	0.00%
50	20%	877.4	21	877.4	877.4	25	0.00%	877.4	76	0.00%
75	10%	1244.1	61	1232.3	1234.9	68	0.96%	1229.6	149	1.18%
75	15%	1254.3	55	1248.8	1256.7	60	0.44%	1248.8	145	0.44%
75	20%	1267.5	50	1267.5	1269.5	56	0.00%	1267.5	139	0.00%
100	10%	1646.4	130	1629.6	1644.8	135	1.03%	1622.9	301	1.45%
100	15%	1689.9	131	1675.3	1687.5	133	0.87%	1675.3	296	0.87%
100	20%	1705.3	102	1698.2	1703.7	108	0.42%	1698.2	289	0.42%
Average		**963.01**	**42**	**959.29**	**962.32**	**23**	**0.25%**	**958.67**	**55**	**0.29%**

[a] PVNS_AMP results from one run only

[b] PVNS_AMP results from 10 runs with different starting seed

[c] PVNS_AMP results from 1 run, fixed iterations

IMP: Improvement on the PVNS_AMP[a]

CPU time in seconds

Average: Solution, Time and IMP

Table 6: Routing Schedule for RVRP without overtime with different light load customers

		Light Load Customers			
	Base Case	**Case 1: 2,11 ∈ L**	**Case 2: 2,10 ∈ L**	**Case 3: 1,10,15∈ L**	**Case 4: 1,10,15,8∈ L**
Routes	0-1-8-3-2-0	0-1-8-3-2-0	0-1-8-3-2-0	0-5-11-16-2-3-1-0	0-5-11-16-2-3-1-0
	0-5-15-10-9-16-11-0	0-5-15-10-9-16-11-0	0-12-15-10-9-11-16-0	0-12-17-15-10-9-0	0-12-17-15-10-9-0
	0-14-20-7-6-0	0-14-20-7-6-0	0-14-20-7-6-0	0-14-20-7-8-6-0	0-14-20-7-8-6-0
	0-18-13-19-4-17-12-0	0-18-13-19-4-17-12-0	0-18-13-19-4-17-5-0	0-18-13-19-4-0	0-18-13-19-4-0
Fleet Composition	3A,1B	3A,1B	3A,1B	3A,1B	3A,1B
Solution	446.16	446.16	476.04	462.32	462.32
TCPU*	2	2	2	<1	<1

*TCPU: total computational time in seconds
L: Light Load customers
Underlined nodes are Light Load Customers

Table 6 shows a computational experiment on the efficient incorporation of the light load customers using RVRP Instance *N = 20*. The first case in Table 6 is the Base Case, which shows the routing schedule when there are no light load customers. A comparison to the base case routing schedule is necessary in order to show the flexibility of the algorithm to incorporate light load customers efficiently, at a minimum extra cost. It can be seen that in the case where customers 2 and 11 are light load, there is no change in the solution structure or the objective function. This is because in the base case, these customers are serviced after the vehicles have become lighter, at the end of their corresponding routes.

When the chosen light load customers are positioned before the light load threshold is reached as in cases 2-4, then an adjustment in the routing is necessary. However, the base routing schedule is mostly preserved in cases 2-4, which suggests that the PVNS_AMP can recognise good quality *Elite Strings* and solution sequences, whilst adjusting for the light load requirement at a very small extra cost.

4.2. The RVRP with Overtime

The mixed integer formulation of the RVRP with overtime provided in Section 3 is tested using Cplex. Our computational experience suggests that the problem is computationally demanding and only one small sized instance is solved to optimality. The RVRP with overtime results from

Cplex are shown in Table 7. The largest instance we solved to optimality is *N = 18*, which is included in Table 7, where instances greater than *N = 50* run out of memory.

Table 7: Cplex results of the RVRP with overtime

N	L (%)	CPLEX				PVNS_AMP			
		LB / Optimal	UB	Fleet Composition	Time[a]	Solution	Fleet Composition	TCPU	Overtime[b]
18	10	**390.3**	-	1A 2B	4	**390.3**	1A 2B	3	7
20	10	413.8	451.1	-	63 [m]	427.2	1A 2B	5	5
20	15	413.8	451.1	-	51 [m]	427.2	1A 2B	5	5
20	20	418.1	448.3	-	84 [m]	427.2	1A 2B	3	5
25	10	474.5	511.8	-	22 [m]	503.1	1A 3B	5	0
30	10	504.9	586.1	-	31 [m]	547.2	4B	7	49
30	15	504.9	586.1	-	3 [m]	547.2	4B	7	49
30	20	503.7	584.7	-	21 [m]	552.6	4B	7	58
50	10	699.1	-	-	32 [m]	820.3	3A 4B	25	17

[a] Computational time in minutes
[b] Overtime used in the solution in minutes
[m] Time recorded until system is Out of Memory
Optimal solutions in bold

It can be seen from Table 7 that in the cases of *N = 30* overtime up to one hour is used, where in other cases, such as *N = 25, L = 10%* no overtime is used at all. The fleet composition and overtime used are consistent across the instances with different percent of light load customers.

The heuristic results from the RVRP with overtime are compared to those without overtime in Table 8. The total cost is provided as well as the fleet composition for each instance and how much overtime is used, if any. Incorporating overtime shows the potential for cost savings up to 8% for one planning period. The saving is not only in terms of overall cost, but also in terms of fleet size. The RVRP proposed in this paper has an interesting characteristic which became apparent during the computational experience. Having allowable overtime and unlimited fleet means that it is very likely that during the search process some candidate solutions could favour an extra vehicle, as opposed to allowing for overtime. Incorporating overtime in advance is also a very important managerial consideration, because vehicle routing is typically characterized with decision making in short term horizons, hence the need for quick and effective decisions. Therefore, exploring greater range of candidate solution structures provides a more comprehensive idea for the possibilities for cost saving.

The instances without overtime are characterized by a larger fleet size. This is because the allowable maximum regular time in some cases restricts the RVRP more tightly than the capacity constraint. That is, a new route is added either when the maximum time is reached or there is no more capacity left in the vehicle. This is an important aspect of the routing in the gas delivery industry, because the time it takes to service a customer (demand-dependent service time) and the travel times (given the lower speed of the vehicles) are quite large. Therefore, considering overtime in advance allows for servicing all customers with fewer vehicles at a lower cost. As a general observation, this finding can be useful in practice when it comes to strategic decisions of buying a new vehicle fleet, and also in daily operations for companies which use hired vehicles or agency drivers.

Table 8: The RVRP results with and without overtime

N	L (%)	PVNS_AMP without overtime		PVNS_AMP with overtime			
		Solution	Fleet Composition	Solution	Total overtime [a]	Fleet Composition	IMP
20	10	446.2	3A 1B	427.2	5	1A 2B	4.42%
20	15	446.2	3A 1B	427.2	5	1A 2B	4.42%
20	20	462.3	3A 1B	427.2	5	1A 2B	7.59%
30	10	560.1	2A 3B	547.2	49	4B	2.30%
30	15	560.1	2A 3B	547.2	49	4B	2.30%
30	20	565.3	2A 3B	552.6	58	4B	2.25%
50	10	852.2	6A 2B	820.3	27	3A 4B	3.74%
50	15	867.2	4A 5B	827.1	36	3A 4B	4.62%
50	20	877.4	6A 4B	842.1	46	3A 4B	4.02%
75	10	1244.1	2A 8B	1230.5	19	4A 6B	1.09%
75	15	1254.3	6A 5B	1241.9	7	2A 8B	0.99%
75	20	1267.5	8A 6B	1253.3	62	3A 7B	1.12%
100	10	1646.4	13A 5B	1549.4	25	3A 10B	5.89%
100	15	1689.9	12A 6B	1579.1	29	3A 11B	6.56%
100	20	1705.3	10A 8B	1592.1	38	3A 11B	6.64%
Average		**963.0**		**924.3**	**30.7**		**3.86%**

IMP: % improvement of the solution when overtime is considered

[a] Overtime used in the solution in minutes

Average: Solution, Overtime and IMP

Table 9: RVRP at a glance Instance *N = 50*

	RVRP without Overtime	**RVRP with Overtime**		
Light Load Customers	**Base Case L = Ø**	**Case 1: L = Ø**	**Case 2: L = 1,5,7,12,9**	**Case 3: L = 1,4,5,7,12,9,32,42,45,50**
Routing	0-6-***24-43-40-7-23-48***-0	0-1-22-28-31-26-8-48-27-0	0-6-14-***24-43-40-7-23-48***-27-0	0-6-14-***24-43-40-7-32-48***-27-0
	0-14-25-13-18-0	*0-**32-2-20***-35-36-3-0	0-8-26-31-28-1-0	0-8-26-31-28-1-0
	0-22-28-31-26-8-27-0	0-6-*23-7-**40-43-24***-25-14-0	0-22-3-36-35-***20-2-32***-46-0	0-22-3-36-35-***20-2-32***-0
	0-11-38-46-0	0-***15-45-33-39-10***-49-5-46-0	0-11-16-29-21-34-50-9-49-5-12-0	0-11-16-29-21-50-34-30-9-49-38-0
	0-1-3-36-35-***20-2-32***-0	0-11-16-29-21-50-34-30-9-38-0	0-38-30-***10-39-33-45-15***-0	0-***10-39-33-45-15***-5-0
	0-9-30-34-50-21-29-16-0	0-37-44-42-19-41-13-18-0	0-46-37-17-***4-47-0***	0-17-37-44-42-19-41-4-12-0
	0-5-49-***10-39-33-45-15***-37-0	***0-47-4***-17-12-0	0-44-42-19-41-13-25-18-0	0-47-18-13-25-0
	0-12-17-44-42-19-41-***4-47-0***	-	-	-
Fleet Composition	**5A, 3B**	**1A, 6B**	**3A, 4B**	**3A, 4B**
AT[a]	**360.2**	**379.6**	**398.3**	**410.1**
AVC[b]	**0.31%**	**0.35%**	**0.35%**	**0.35%**
AL[c]	**14116**	**15149**	**15828**	**16132**
Overtime[d]	**0**	**10**	**27**	**46**
Solution	**848.3**	**825.6**	**820.3**	**842.1**

Underlined nodes are light load customers, nodes in bold are *Elite Strings*

[a] AT is Average travel time per vehicle in minutes

[b] AVC is Average variable cost per vehicle as a proportion of total cost

[c] AL is Average load per vehicle

[d] Overtime used in minutes in the solution

We have shown that there is an opportunity of cost savings when light load customers are incorporated into the routing schedule and also if overtime is incorporated in advance. However, we also show the combined effect of the real life attributes in Table 9, which portrays Instance *N = 50* of the RVRP with and without overtime and with different light load customers. The key observations are summarized below.

Managerial Insights

Effects of Light Load: Similar to our findings from Table 6, here we can also observe that the route composition is mostly preserved regardless of the overtime and the light load customer composition. Moreover, the increase of the objective function from Case 1 to Case 3 is only 1.99%, which has 20% light load customers and 46 minutes of overtime. This means that the PVNS_AMP is flexible enough to identify and preserve good quality *Elite Strings* in a consistent manner, which are relevant for all attributes of the RVRP at a very small extra cost.

Effects of Overtime: The examples with overtime of the RVRP tend to favour the larger vehicle type B, which results in a smaller fleet size. Here an interesting observation is that even though the fleet mix is composed of more vehicles of type B (larger vehicles), the average variable cost of travel remains unchanged. Additionally, when considering overtime, the vehicle capacity is 12.5% better utilized, because the average load carried by the fleet is greater when overtime is considered in advance. Moreover, the working time is 12% better utilized, as the average travel time of the fleet is higher and much closer to the maximum allowable regular time. This is an important managerial consideration in relation to drivers working hours' directive and the effective management of human resources. The minimum number of customers per route is 4, hence there are no short routes. This suggests that incorporating overtime contributes to a better utilization of the drivers' time, especially if they are in full time employment.

Combined Effect: Another interesting observation is the combined effect of having light load customers and allowable overtime. It can be seen that for the Base Case, the objective function is 825.6, with 5 vehicles of type B and only 1 of type A. In contrast Case 2 has an objective function of 820.3. This means that having light load customers can actually improve the efficiency of the routing when overtime is allowed. It provides an opportunity for servicing more light load customers on a given route after the maximum regular time when there is still capacity left, rather

than placing them on a different route. These findings are not only relevant to the instance portrayed in Table 9, but also across the other test instances in this study.

Overtime vs Fleet Mix Trade-off: This aspect became apparent during our computational experience. Having allowable overtime coupled with unlimited fleet results in an interesting trade-off between servicing more customers in the overtime or having an extra vehicle. Looking at the Base Case in Table 9 there are a total of 8 vehicles needed to satisfy the total demand, whereas in Case 1 where overtime is allowed we only need 7 vehicles. Even though the fleet mix is different, the total cost is lower. This particular aspect can be applied in practice for medium to long term strategic planning, when companies decide to buy or replace their own fleet. In daily operations it can also be useful if a company has a mix of owned and hired fleet, or a mix of full time and agency drivers.

4.3. Special case of the RVRP: The Fleet Size and Mix VRP

Similar to most heuristic methods, the solution methods designed to solve RVRPs are problem specific. Typically, they are not tested on well-known literature benchmark instances, because one cannot directly compare methods designed for different problems. However, we test the PVNS_AMP on the well-known literature benchmark instances by Golden et al. (1984), with fixed cost and variable cost. Moreover, we test our algorithm on large scale VRP instances with Heterogeneous fixed fleet by Li, Golden, Wasil (2007). The results from the computational experiments are shown in tables 10 - 12, and are compared to relevant heuristic methods, as well as the Best Known Solutions (BKS) from the literature, with the respective average optimality gap**.** We have used the PVNS_AMP version with a stopping criterion of 20 fixed iterations. It can be seen from the tables that the proposed PVNS_AMP can be successfully applied to the different versions of the FSMVRP, as well as to the HFVRP. Even though the method is primarily designed for a RVRP with specific real life attributes, it shows competitive performance, yielding less than 0.03% average deviation from the BKS for the FSMVRP with up to 100 customers and 1% on the HFVRP with up to 360 customers. Compared to some non-exact methods such as GA and TS, the computational time of our method is competitive. We show both average running times, as reported by the respective authors, as well as corresponding scaled average times, which we adjusted for the differences in PC performance against ours, where available, using the comparison website http://cpuboss.com/compare-cpus.

Table 10: Results on Golden et al. (1984) FSMVRP instances with fixed cost

Instance	N	BKS	TSA1[d]		ILS-RVND-SP[e]		GA[f]		PVNS_AMP	
			Sol	Time	Sol	Time	Sol	Time	Sol	Time
3	20	961.03[abc]	**961.03**	21	**961.03**	0	**961.03**	21	**961.03**	32
4	20	6437.33[abc]	**6437.33**	22	**6437.33**	0	**6437.33**	18	**6437.33**	29
5	20	1007.05[abc]	**1007.05**	20	**1007.05**	0	**1007.05**	13	**1007.05**	28
6	20	6516.47[abc]	**6516.47**	25	**6516.47**	0	**6516.47**	22	**6516.47**	31
13	50	2406.36[abc]	**2406.36**	145	**2406.36**	2	**2406.36**	91	**2406.36**	65
14	50	9119.03[abc]	**9119.03**	220	**9119.03**	2	**9119.03**	42	**9119.03**	56
15	50	2586.37[abc]	2586.84	110	**2586.37**	6	**2586.37**	48	**2586.37**	52
16	50	2720.43[abc]	2728.14	111	**2720.43**	4	2724.22	107	**2720.43**	55
17	75	1734.53[b]	1736.09	322	**1734.53**	12	**1734.53**	109	**1734.53**	99
18	75	2369.65[ab]	2376.89	267	**2369.65**	12	**2369.65**	197	**2369.65**	124
19	100	8661.81[b]	8667.26	438	**8661.81**	25	8662.94	778	8667.26	269
20	100	4032.81	4048.09	601	**4032.81**	46	4038.45	1004	4038.45	237
Average			**0.09%**	**192**	**0.00%**	**9**	**0.02%**	**204**	**0.02%**	**89**
Scaled Average Time				**-**		**3**		**74**		**89**

[a] Optimality proven by Pessoa, Uchoa, Poggi (2009)

[b] Optimality proven by Baldacci, Mingozzi (2009)

[c] Optimality proven by Choi and Tcha (2007)

[d] Brandao (2009)

[e] Subramanian et al. (2012)

[f] Liu, Huang, Ma (2009)

Time: in seconds

Average: Gap and Time

Scaled average time: in seconds adjusted for PC specifications where available

Table 11: Results on Golden et al. (1984) FSMVRP instances with variable cost

Instance	N	BKS	VNS1[d]		ILS-RVND[e]		GA[f]		PVNS_AMP	
			Sol	Time	Sol	Time	Sol	Time	Sol	Time
3	20	623.22[abc]	**-**	-	**623.22**	4	-	**-**	**623.22**	35
4	20	387.18[abc]	**-**	-	**387.18**	3	-	**-**	**387.18**	32
5	20	742.87[abc]	**-**	-	**742.87**	5	-	**-**	**742.87**	36
6	20	415.03[abc]	**-**	-	**415.03**	3	-	**-**	**415.03**	28
13	50	1491.86[abc]	**1491.86**	310	**1491.86**	31	**1491.86**	**117**	**1491.86**	69
14	50	603.2[abc]	**603.2**	161	**603.2**	14	**603.2**	**26**	**603.2**	58
15	50	999.8[abc]	**999.8**	218	**999.8**	15	**999.8**	37	**999.8**	63
16	50	1131[abc]	**1131**	239	**1131**	17	**1131**	54	**1131**	61
17	75	1038.6[abc]	**1038.6**	509	**1038.6**	48	**1038.6**	153	**1038.6**	142
18	75	1800.8[ab]	**1800.8**	606	**1800.8**	53	1801.4	394	1801.4	121
19	100	1105.44[bc]	**1105.44**	1058	**1105.44**	78	**1105.44**	479	**1105.4**	201
20	100	1530.43[bc]	1533.24	1147	1530.52	87	1534.37	826	1534.37	213
Average			**0.02%**	**531**	**0.00%**	**30**	**0.02%**	**261**	**0.02%**	**88**
Scaled Average Time				**-**		**11**		**85**		**88**

[a] Optimality proven by Pessoa, Uchoa, Poggi (2009)

[b] Optimality proven by Baldacci and Mingozzi (2009)

[c] Optimality proven by Choi and Tcha (2007)

[d] Imran, Salhi, Wassan (2009)

[e] Penna, Subramanian, Ochi (2011)

[f] Liu, Huang, Ma (2009)

Time is in seconds

Average: Gap and Time

Scaled average time: in seconds adjusted for PC specifications where available

Table 12: Results on Li, Golden, Wasil (2007) HFVRP instances with variable cost

Instance	N	BKS	HRTR[a]		TSA[b]		ILS-RVND-SP[c]		PVNS_AMP	
			Sol	Time	Sol	Time	Sol	Time	Sol	Time
H1	200	12050.08	12067.65	688	**12050.08**	1395	**12050.08**	72.1	**12050.08**	1023
H2	240	10208.32[d]	10234.4	995	10226.17	3650	10329.15	176.43	10295.36	2698
H3	280	16223.39[d]	16231.8	1438	16230.21	2822	16282.41	259.61	16305.21	3152
H4	320	17458.65	17576.1	2256	**17458.65**	8734	17743.68	384.52	17761.9	5469
H5	360	23166.56[d]	-	-	23220.72	13,321	23493.87	621.17	23612.23	8554
Average			**0.28%**	**1344**	**0.09%**	**5984.4**	**0.92%**	**303**	**1.00%**	**4179**
Scaled Average Time				**-**		-		**116**		**4179**

[a]Li et al. (2007)

[b]Brandão (2011)

[c]Subramanian et al. (2012)

[d]Found by Brandão (2011) with different TSA calibration

Time is in seconds

Average: Gap and Time

Scaled average time: in seconds adjusted for PC specifications where available

All of the algorithms we compare against, reported in tables 10 - 12 are coded in C or C++ (except for Li, Golden, Wasil (2007), which is not specified). These have very similar performances to ours. However, the algorithms have been tested and programmed on different machines and some use different operating systems. For instance, Subramanian et al. (2012) and Penna, Subramanian, Ochi (2011) used Intel Core i7 with 2.93GHz, Imran, Salhi and Wassan (2009) Intel Pentium M 1.7 GHz and Liu, Huang and Ma (2009) Intel Pentium 4, 3 GHz. Li, Golden, Wasil (2007) AMD Athlon 1 GHz and Brandão (2011) Intel Pentium M 1.4 GHz. Though not all machines can be compared consistently using one standard CPU benchmarking source, we opted for the comparison website http://cpuboss.com/compare-cpus. The website generates, where possible, an overall score out of 10 for the machines in question, which reflects their relative performance capability.

5. Conclusion

The subject of this paper is a real life routing problem which arises in the gas delivery industry, characterized by heterogeneous fleet, demand-dependent service times, maximum allowable overtime and light loads. We present a mathematical formulation, which is tested on Cplex and optimal solutions and lower / upper bounds are achieved where possible. We have also developed a new learning-based algorithm which uses memory structures embedded in a Population VNS. The computational experience suggests that the learning mechanisms based on Adaptive Memory can improve the performance of the PVNS with up to 5.2% when applied to the RVRP. The use of *Elite Strings* as a main driver of memory exploitation, results in the recognition of good solution sequences which can guide the search process towards better regions of the solution topography. Moreover, it shows that memory structures can be used with a powerful memoryless metaheuristic method, as long as an appropriate mechanism to recognise good solution sequences is in place. The performance of the PVNS_AMP is empirically tested and analysed, and it is compared to the solutions achieved by Cplex, as well as standard literature benchmark instances.

The findings show that the routing efficiency can be improved significantly when light load customers and overtime are considered in advance. On average, there are possible savings for practitioners with up to 8% in the daily routing cost. Moreover, a better fleet utilization in terms of vehicle capacity, as well as a better utilization of the drivers' working hours can be achieved with up to 12.5% and 12% respectively. We believe that further research on problems with light

load requirement and allowable overtime can be triggered from our findings, as well as further research on the hybridization of the AMP rationale with different iterative heuristic methods.

Acknowledgements

The first author is grateful to the Kent Business School for the PhD funding and support.

References

Archetti C., Savelsbergh M., Speranza M. (2016) The Vehicle Routing Problem with Occasional Drivers, European Journal of Operational Research 254, 472–480.

Baldacci R., Mingozzi A. (2009) A unified exact method for solving different classes of Vehicle Routing Problems, Mathematical Programming 120 (2), 347–380.

Battarra M., Monaci M., Vigo D. (2009) An adaptive guidance approach for the heuristic solution of a minimum multiple trip Vehicle Routing Problem, Computers & Operations Research 36, 3041–3050.

Brandão J. (2009) A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem, European Journal of Operational Research 195, 716–728.

Brandão, J. (2011) A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem, Computers & Operations Research 38, 140–151.

Choi E., Tcha D. (2007) A column generation approach to the heterogeneous fleet Vehicle Routing Problem, Computational Operations Research 34, 2080–2095.

Cornillier F., Laporte G., Boctor F., Renaud J. (2009) The petrol station replenishment problem with time windows. Computers & Operations Research 36(3), 919–935.

Erdogan S., Miller-Hooks E. (2012) A green Vehicle Routing Problem, Transportation Research Part E: Logistics and Transportation Review 48 (1), 100–114.

Gillet B., Miller L. (1974) A heuristic algorithm for the vehicle dispatch problem, Operations Research 22, 340–349.

Goel A., Gruhn V. (2008) A General Vehicle Routing Problem, European Journal of Operational Research 191, 650–660.

Golden B., Assad A, Levy L, Gheysens F. (1984) The fleet size and mix Vehicle Routing Problem, Computers & Operations Research 11, 49–66.

Gribkovskaia I., Gullberg B.O., Hovden K.J., Wallace S.W. (2006) Optimization model for a livestock collection problem, International Journal of Physical Distribution & Logistics Management 36, 136–52.

Hasle G., Løkketangen A., Martello S. (2006) Rich models in discrete optimization: Formulation and resolution, European Journal of Operational Research, vol. 175, pp. 1752–1753.

Imran A., Salhi S., Wassan N. (2009) A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem, European Journal of Operational Research, vol. 197(2), pp. 509–518.

Kalayci C., Kaya B. (2016) An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery, Expert Systems with Applications 66, 163–175.

Kok A.L., Hans E.W., Schutten J.M.J. (2012) Vehicle routing under time-dependent travel times: The impact of congestion avoidance, Computers & Operations Research 39, 910–918.

Lahyani R., Khemakhem M., Semet F. (2015) Rich Vehicle Routing Problems: From a taxonomy to a definition, European Journal of Operational Research 241, 1–14.

Li, F., Golden, B., Wasil E. (2007) A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem, Computers & Operations Research vol. 34, pp. 2734–2742.

Li X., Tian P., Aneja Y.P. (2010) An Adaptive Memory programming metaheuristic for the heterogeneous fixed fleet Vehicle Routing Problem, Transportation Research Part E 46, 1111–1127.

Li X., Leung S., Tian P. (2012) A multistart Adaptive Memory-based tabu search algorithm for the heterogeneous fixed fleet open Vehicle Routing Problem, Expert Systems with Applications 39, 365–374.

Liu S., Huang W., Ma H. (2009) An effective genetic algorithm for the fleet size and mix Vehicle Routing Problem, Transportation Research Part E 45, 434–445.

Matei O., Pop P.C., Sas I. and Chira C. (2015) An improved immigration memetic algorithm for solving the heterogeneous fixed fleet vehicle routing problem, Neurocomputing, 150, 58–66.

Mladenovic N., Hansen P. (1997) Variable Neighbourhood search, Computers & Operations Research 24, 1097–1100.

Mladenovic N., Salhi S., Hanafi S., Brimberg J. (2016) Recent Advances in Variable Neighbourhood Search, Computers & Operations Research 52, 147–148.

Moon I. K., Lee J.H., Seong J. (2012) Vehicle Routing Problem with time windows considering overtime and outsourcing vehicles, Expert Systems with Applications 39, 13202–13213.

Naji-Azimi Z., Salari M., Renaud J., Ruiz A. (2016) A practical Vehicle Routing Problem with desynchronized arrivals to depot, European Journal of Operational Research 255, 58–67.

Nagy G., Wassan NA., Salhi S. (2013) The vehicle routing problem with restricted mixing of deliveries and pickups, Journal of Scheduling 16 (2), 199–213.

Oppen J., Loketangen A. (2008) A tabu search approach for the livestock collection problem, Computers & Operations Research 35, 3213–3229.

Ren Y., Dessouky M., Ordonez F. (2010) The multi-shift Vehicle Routing Problem with overtime Computers & Operations Research 11, 1987–1998.

Penna P., Subramanian A., Ochi L. (2011) An iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem, Journal of Heuristics 19 (2), 201–232.

Pessoa A., Uchoa E., Poggi de Aragão M. (2009) A robust branch-cut-and-price algorithm for the heterogeneous fleet Vehicle Routing Problem, Networks 54, 167–177.

Rochat Y., Taillard E.D. (1995) Probabilistic diversification and intensification in local search for vehicle routing, Journal of Heuristics 1, 147–167.

Salhi S. (2017) Heuristic search: The emerging science of problem solving, Cham, Switzerland: Palgrave MacMillan, Spring International Publishing AG.

Seixas M.P., Mendes A.B. (2013), Column generation for a multitrip Vehicle Routing Problem with time-windows, driver work hours, and heterogeneous fleet, Mathematical Problems in Engineering, 1–13.

Stenger A., Vigo D., Enz S., Schwind M. (2013) An adaptive Variable Neighbourhood Search Algorithm for a Vehicle Routing Problem arising in small package shipping, Transportation Science 47, 64–80.

Subramanian A., Penna P., Uchoa E., Ochi L. (2012) A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem, European Journal of Operational Research 221, 285–295.

Sze J.F., Salhi S., Wassan N. (2016) A hybridisation of adaptive variable neighbourhood search and large neighbourhood search: Application to the vehicle routing problem, Expert Systems with Applications 65, 383–397.

Sze J.F., Salhi S., Wassan N. (2017) The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search, Transportation Research Part B 101, 162–184.

Tarantillis C.D., Kiranoudis C.T. (2002) BoneRoute: An Adaptive Memory-based method for effective fleet management, Annals of Operations Research 115, 227–241.

Tarantillis C.D. (2005) Solving the Vehicle Routing Problem with Adaptive Memory programming methodology, Computers & Operations Research 32, 2309–2327.

Vidal T., Crainic T.G., Gendreau M., Prins C. (2014) A unified solution framework for multi-attribute Vehicle Routing Problems, European Journal of Operational Research 234, 658–673.

Wassan Naveed., Wassan N., Nagy G., Salhi S. (2017) The multiple trip Vehicle Routing Problem with backhauls: Formulation and a two-level Variable Neighbourhood search, Computers & Operations Research 78, 454–467.

Yin PY., Glover F., Laguna M. (2010) Cyber Swarm Algorithms – Improving particle swarm optimization using Adaptive Memory strategies, European Journal of Operational Research 201, 377–389.

Zachariadis E., Tarantilis C., Kiranoudis C. (2015) The load-dependent vehicle routing problem and its pick-up and delivery extension, Transportation Research Part B 71, 158–181.